Capstone Project – 3

Cardiovascular Risk Prediction

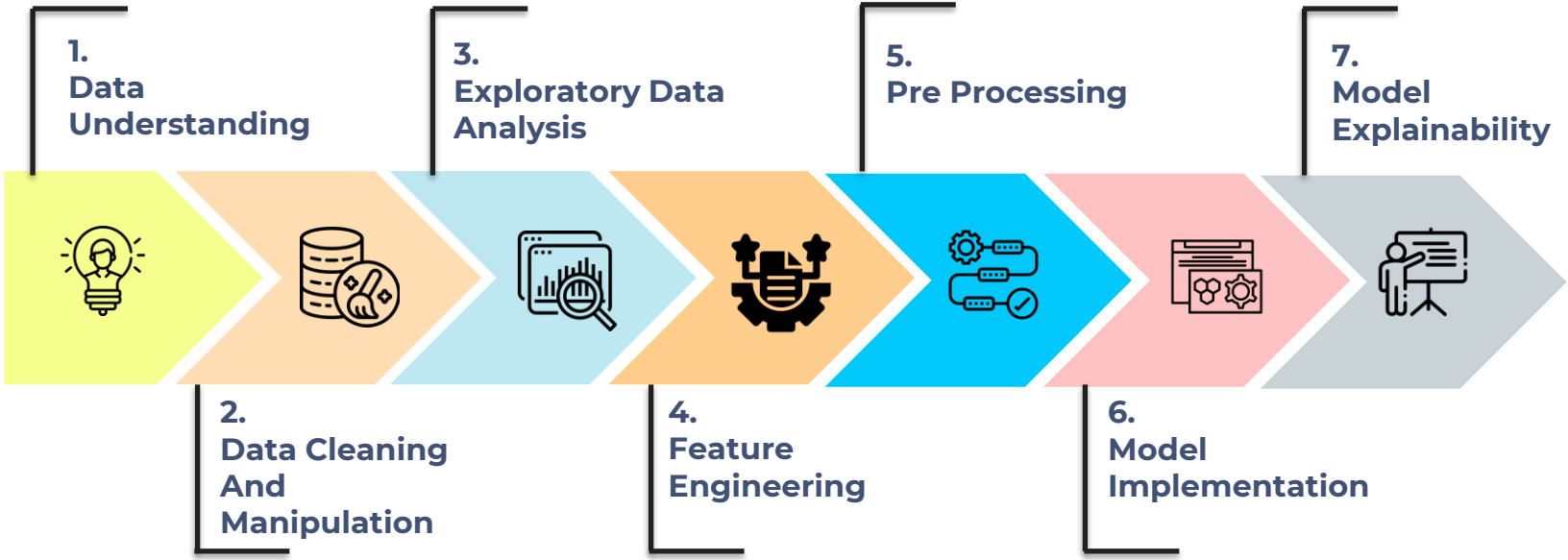
By

SHAIK AHMAD BASHA

Problem Statement

The dataset is from an ongoing cardiovascular study on residents of the town of Framingham, Massachusetts. The classification goal is to predict whether the patient has a 10-year risk of future coronary heart disease (CHD). The dataset provides the patients' information. It includes over 4,000 records and 15 attributes. Each attribute is a potential risk factor. There are both demographic, behavioral, and medical risk factors.

Work Flow:


1.
Data
Understanding
2.
Data Cleaning
And
Manipulation
3.
Exploratory Data
Analysis
4.
Feature
Engineering
5.
Pre Processing
6.
Model
Implementation
7.
Model
Explainability

Data Understanding :

The dataset has 3390 rows and 17 columns. It contains patient's health information like demographic, behavioral, and medical risk factors.

The features of the dataset are :

Demographic:

- **sex : male or female**
- **Age: Age of the patient**

Behavioral

- **is_smoking: whether or not the patient is a current smoker ("YES" or "NO")**
- **Cigs Per Day: the number of cigarettes that the person smoked on average in one day.**

Medical(history)

- **BP Meds: whether or not the patient was on blood pressure medication.**
- **Prevalent Stroke: whether or not the patient had previously had a stroke.**
- **Prevalent Hyp: whether or not the patient was hypertensive.**
- **Diabetes: whether or not the patient had diabetes.**

Medical(current)

Tot Chol: total cholesterol level.

Sys BP: systolic blood pressure.

Dia BP: diastolic blood pressure

BMI: Body Mass Index.

Heart Rate: heart rate.

Glucose: glucose level.

Predict variable (desired target)

10-year risk of coronary heart disease CHD(binary: “1”, means “Yes”, “0” means “No”)

Data Cleaning and Manipulation :

In Data cleaning and manipulation, we will check for null values, duplicated values and manipulate the data for our need.

In the data there are 7 features with null values. They are

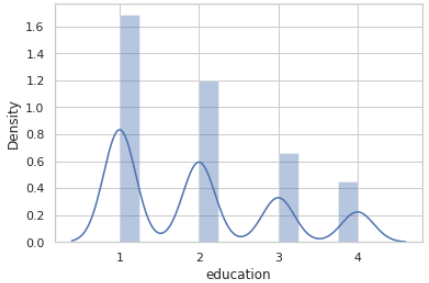
- education
- cigsPerDay
- BPMeds
- totChol
- BMI
- heartrate
- Glucose

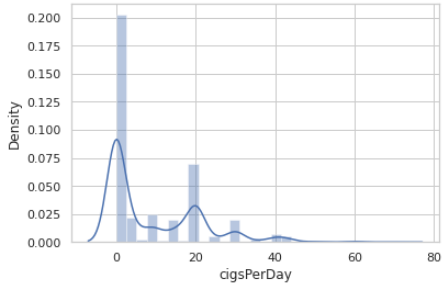
Before handling the null values of those feature, we have observe the distribution of values of those features.

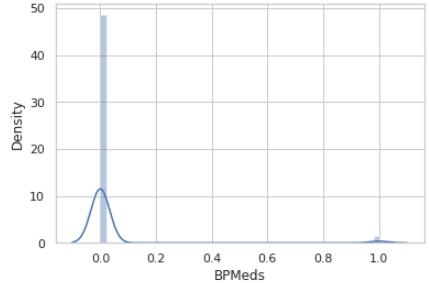
```
# Checking for null values
data.isnull().sum()
```

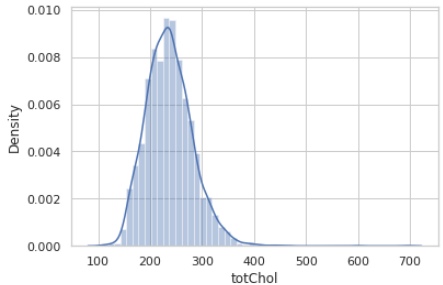
```
id                   0
age                  0
education           87
sex                  0
is_smoking           0
cigsPerDay          22
BPMeds              44
prevalentStroke      0
prevalentHyp         0
diabetes             0
totChol             38
sysBP                0
diaBP                0
BMI                 14
heartRate            1
glucose            304
TenYearCHD           0
dtype: int64
```

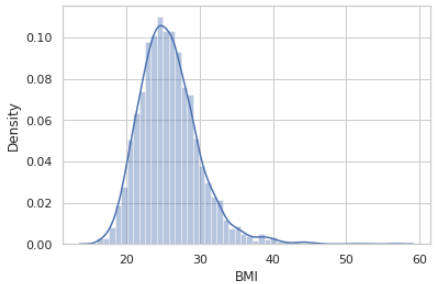
Distribution of features with null values.

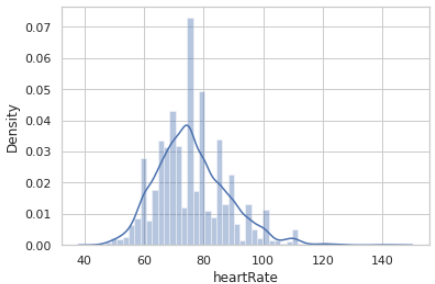


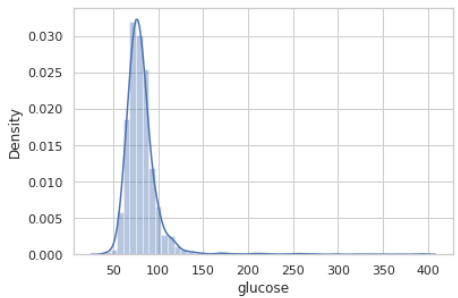












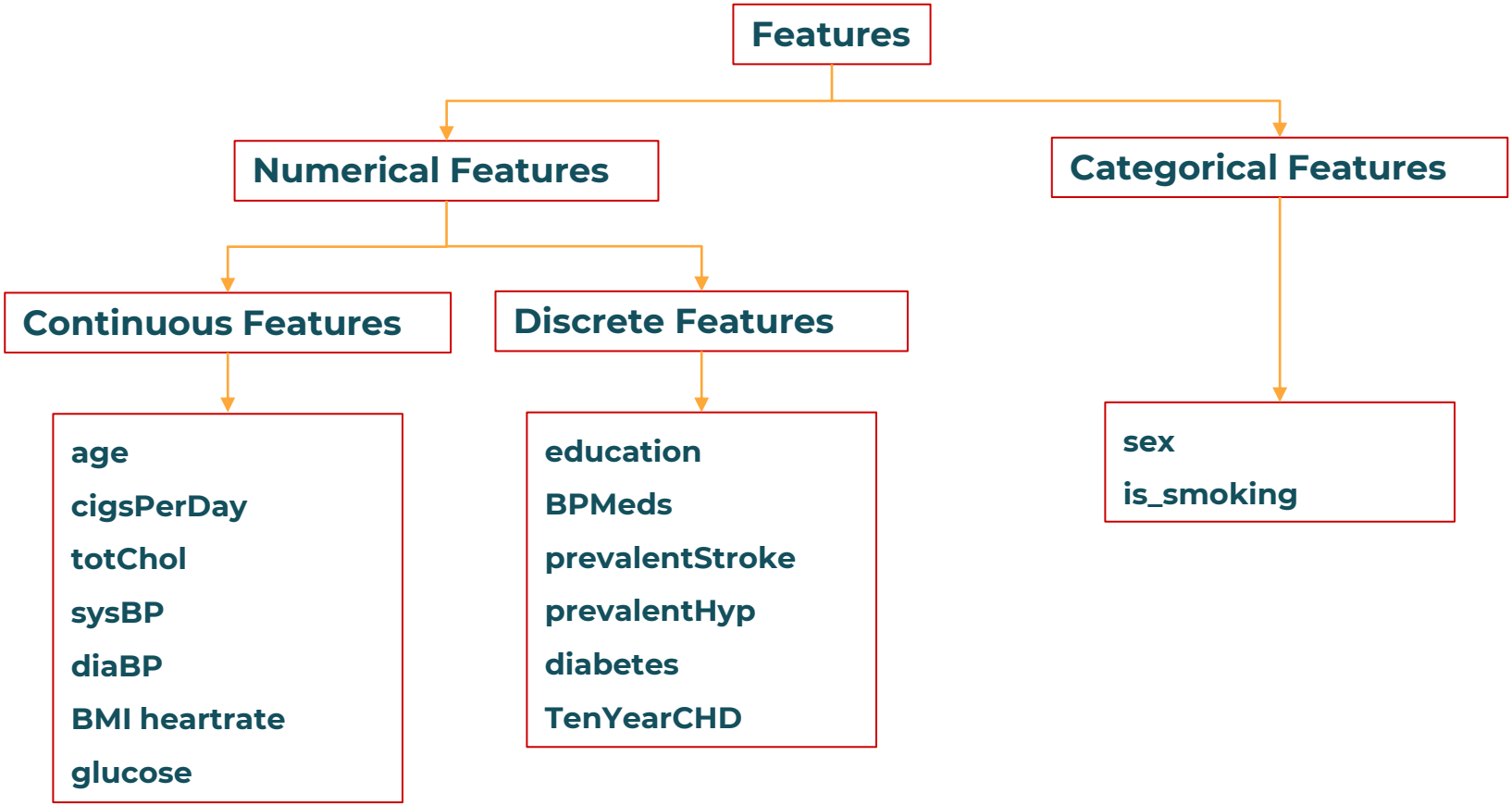
From the above distribution plots, there are few categorical features exists in numerical form.

- education, BPMeds are categorical features
- cigsPerDay, totCHol, BMI, heartRate, glucose are numerical features.

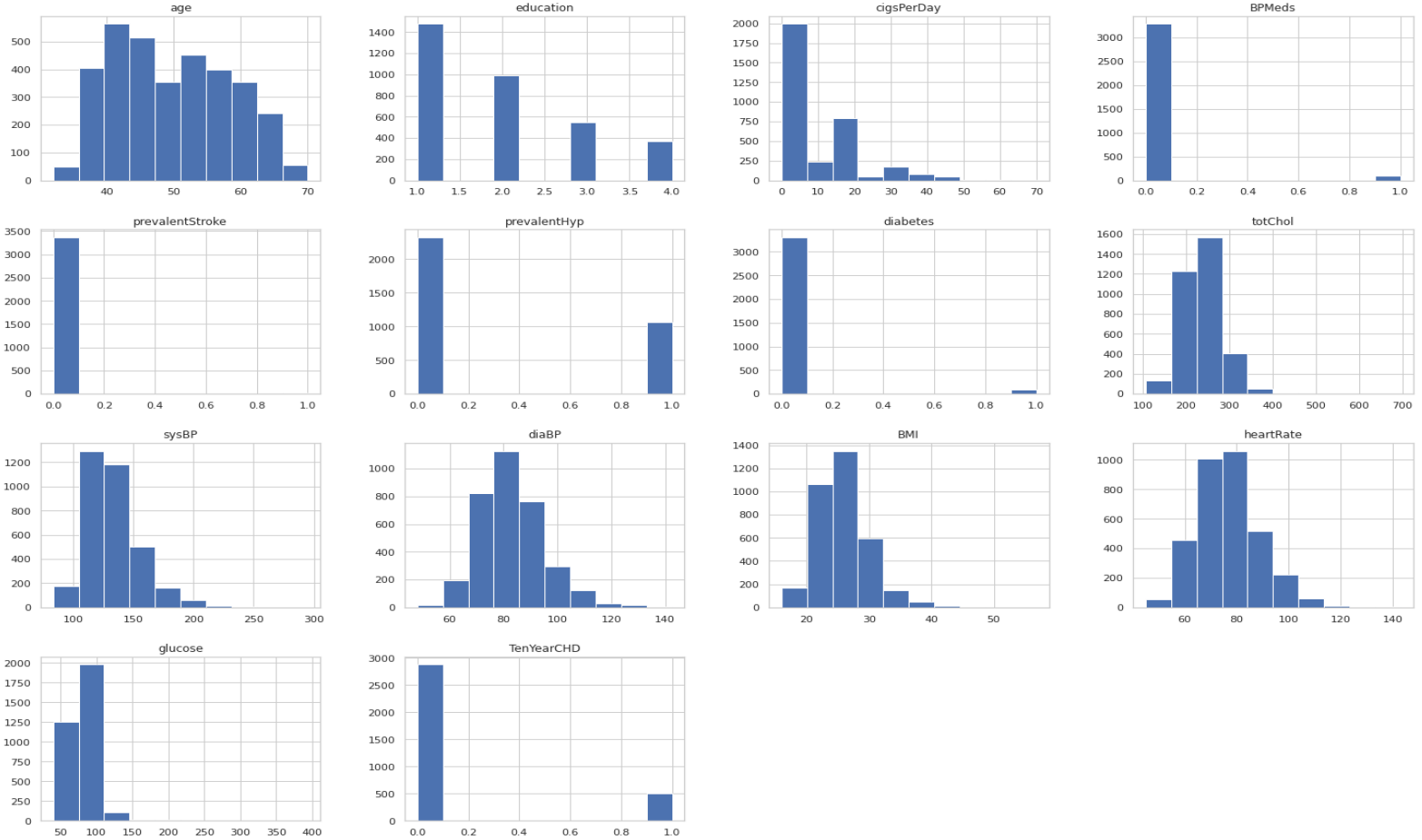
- All the above numerical features is skewed towards right side. Hence we can replace null values of those features with median values.
- When it come to handle the null values for categorical features, we can replace the null values with most frequent value i.e, mode

```
# Handling Null Values

# For Numerical Features
data['cigsPerDay'].fillna(data['cigsPerDay'].median(), inplace = True)
data['totChol'].fillna(data['totChol'].median(), inplace = True)
data['BMI'].fillna(data['BMI'].median(), inplace = True)
data['heartRate'].fillna(data['heartRate'].median(), inplace = True)
data['glucose'].fillna(data['glucose'].median(), inplace = True)

# For Categorical Features
data['education'].fillna(float(data['education'].mode()), inplace = True)
data['BPMeds'].fillna(float(data['BPMeds'].mode()), inplace = True)
```

Exploratory Data Analysis :



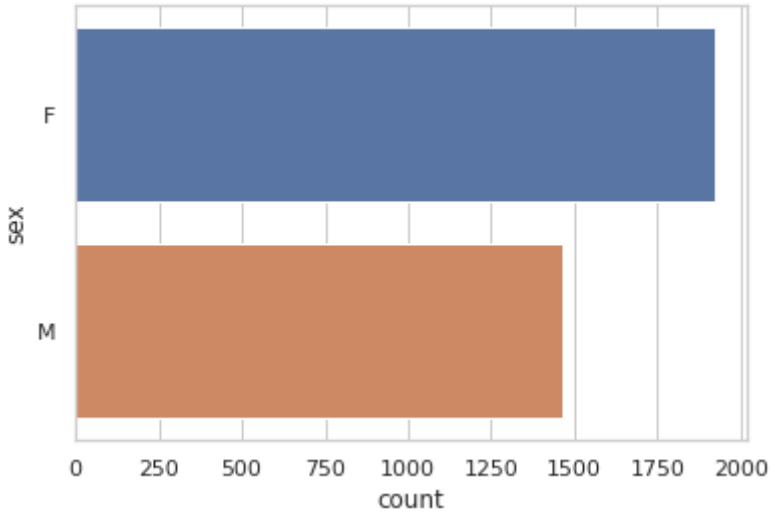
Distribution of numerical features :

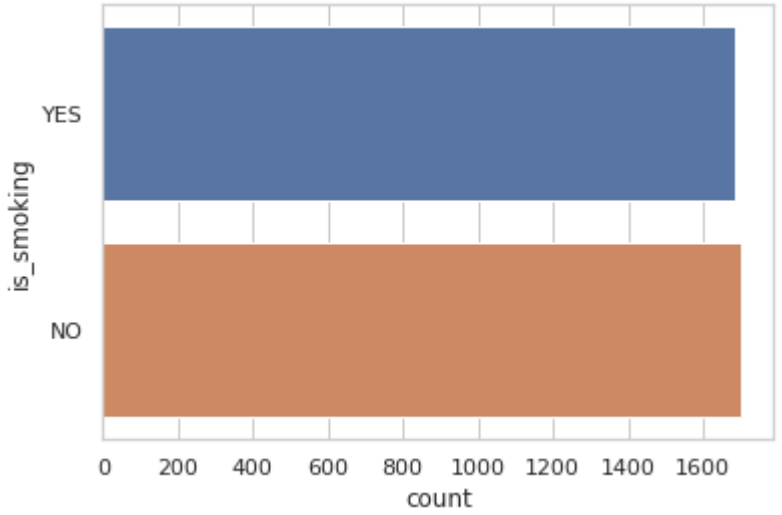


From the above histograms, we can observe that

- Our dependent feature, is actually an indicator variable with only two possible values; 0 and 1
- Some featutres like prevalentStroke, prevalentHyp, diabetes, BPMeds, education and target features should be categorical features.
- Most number of patient's age falls in the range 35 – 45.
- Most number of patients belongs to education category '1.0'.
- Most number of patients smoke cigarette in range of 0 - 10.
- Most number of patients are not on BP Medications and most number of patients does not have a stroke previously .
- Most number of patients are not hypertensive and not diabetic also.
- Most number of patient's Cholesterol falls in the range 200 – 300.
- Most number of patient's Systolic Blood Pressure falls in the range 100 – 150.
- Most number of patient's Diastolic Blood Pressure falls in the range 70 - 90.
- Most number of patient's BMI falls in the range 20 - 30. And a very few has above 40.
- Most number of patient's Heart Rate falls in the range 70 – 90
- Most number of patient's Glucose levels falls in the range 50 - 100.

Distribution of categorical features :

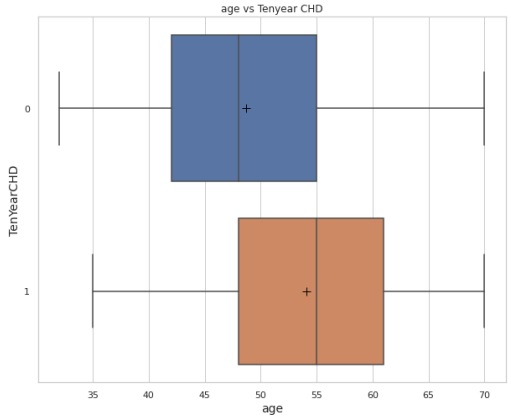


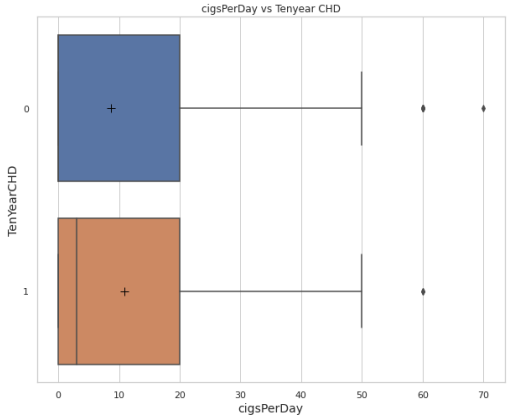


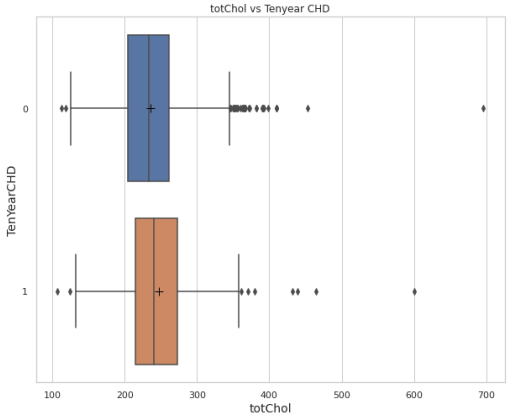
From the above visualizations, we can observe that

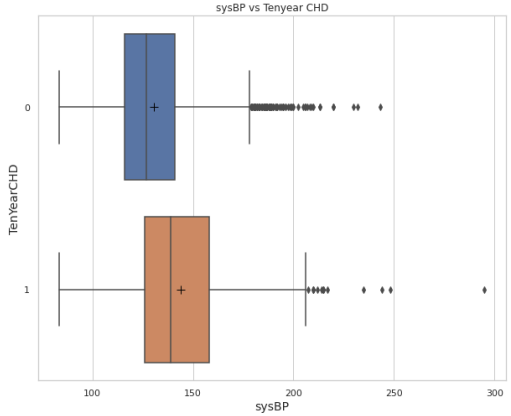
- ❖ **Most of the patients are females**
- ❖ **Almost half of the patients have a habit of smoking**

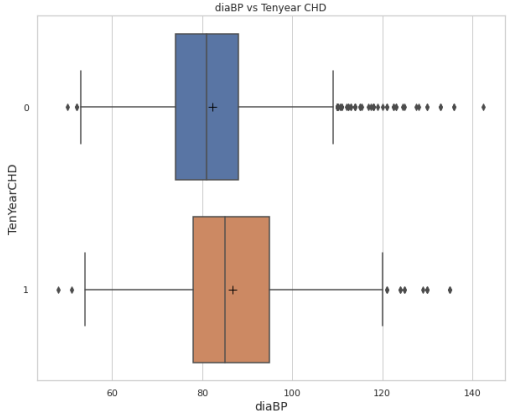
Relation Between Continuous Features and Target Feature :

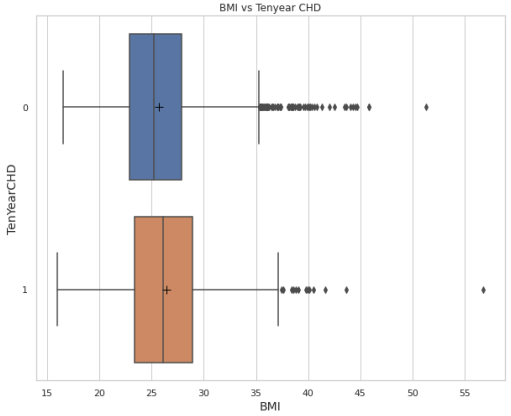


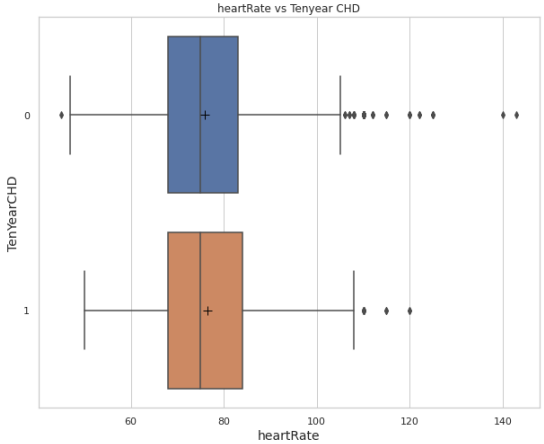


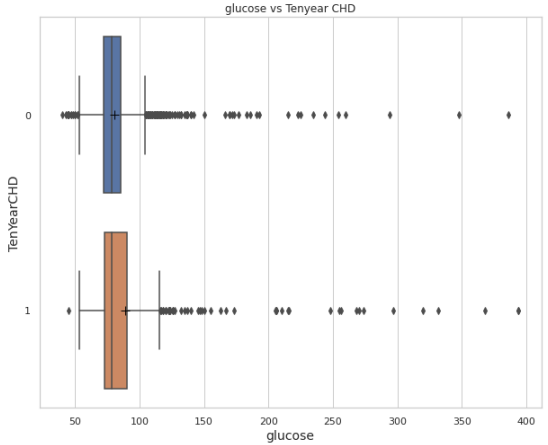








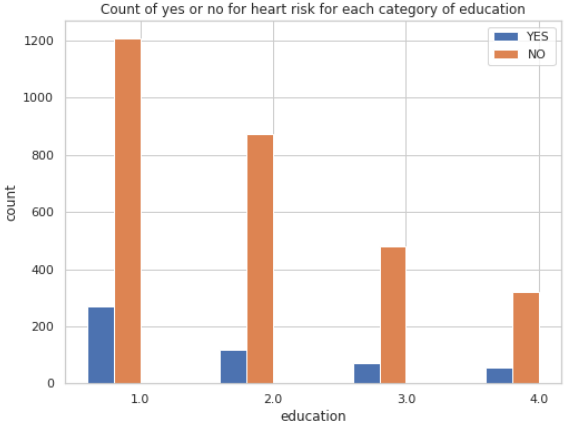


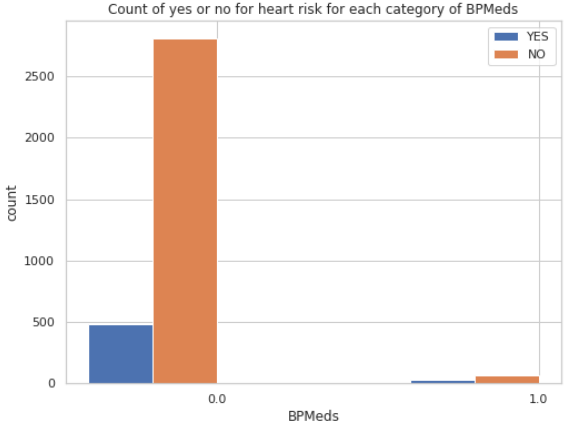


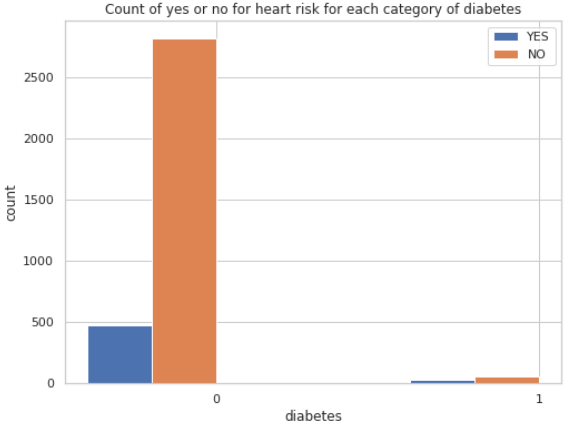
From the above boxplots, we can observe that

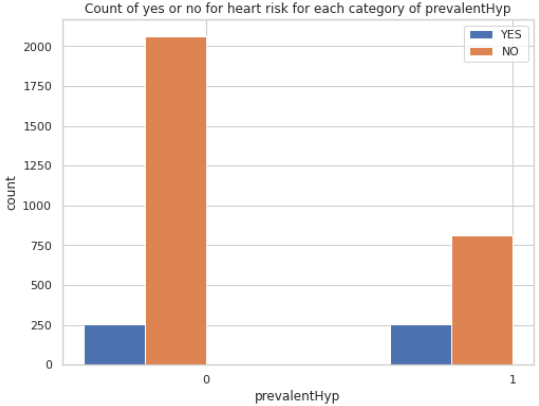
- ❖ **Outliers present in continuous features except age column.**
- ❖ **The feature 'cigsPerDay' is equally distributed, which means the feature is not a good predictor.**
- ❖ **The patients who have more age, totChol, sysBP, diaBP, BMI, heartRate, glucose are tending towards risk of coronary heart disease.**

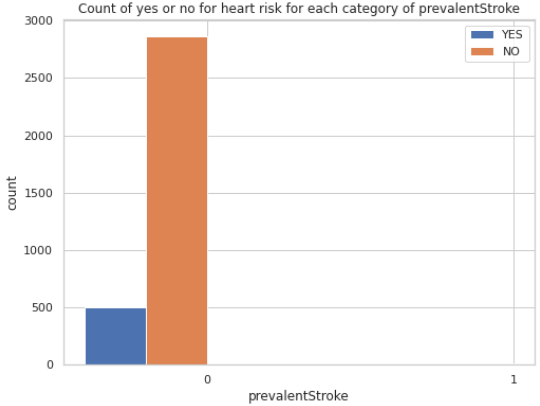
How discrete features distributed among target feature :



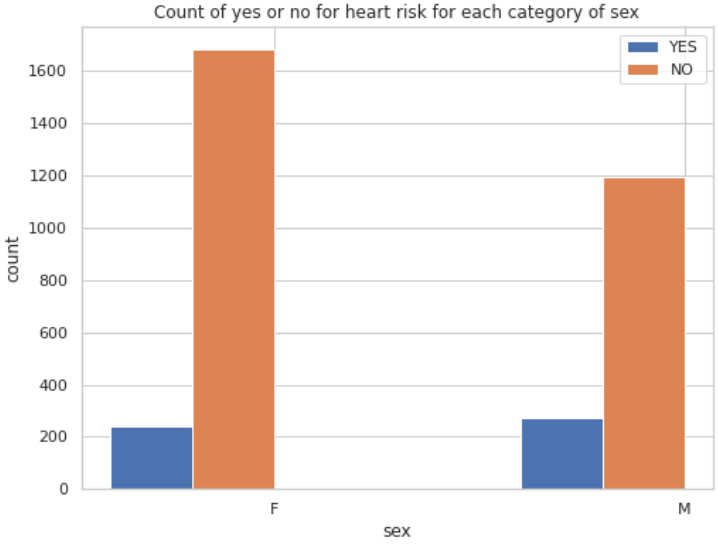


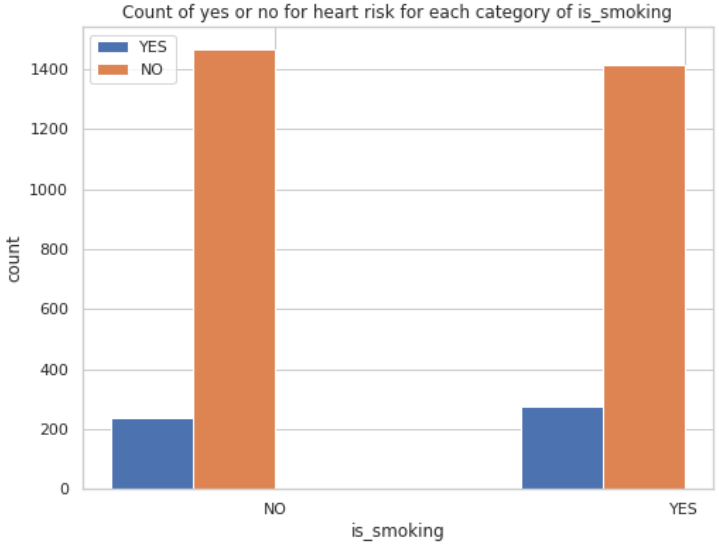






How categorical features distributed among target feature :

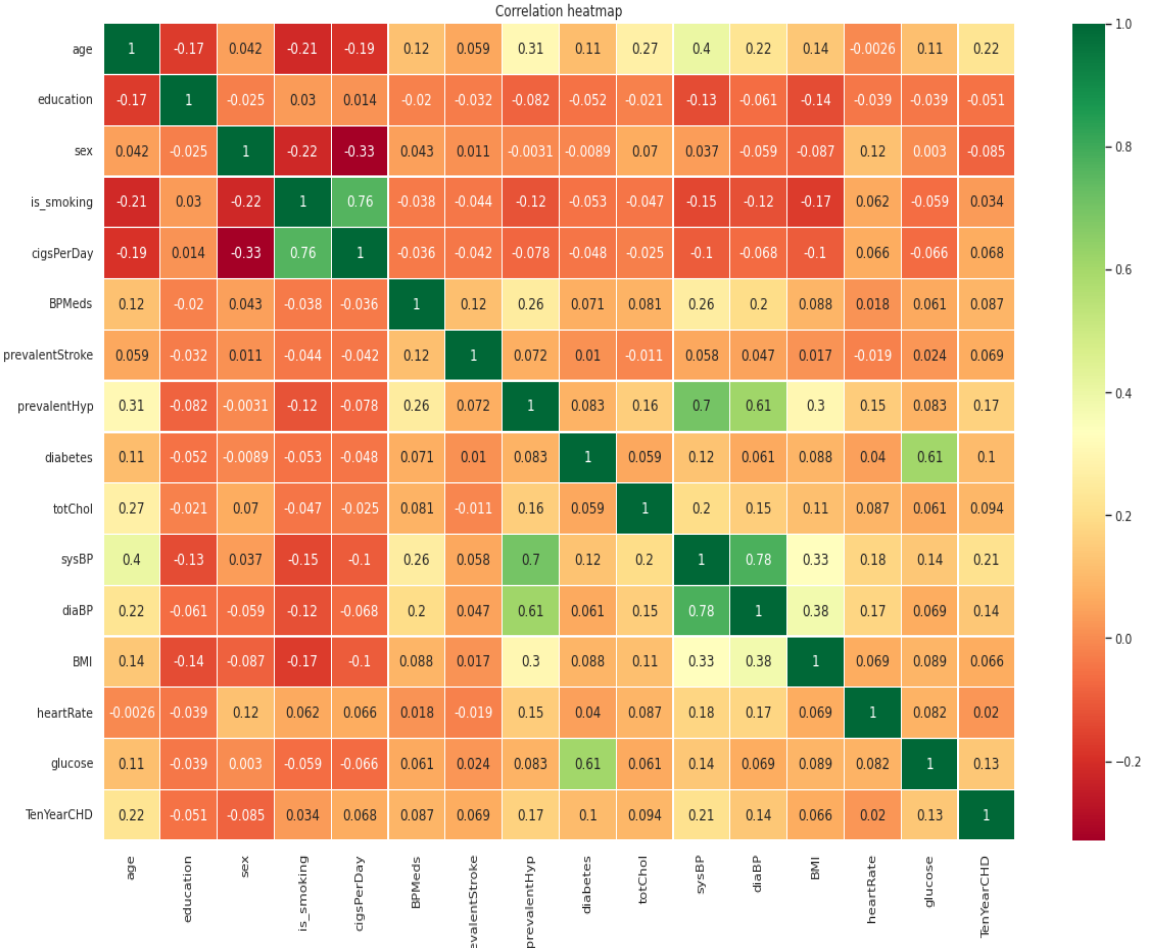




Correlation Heatmap :

Hence, from the above correlation heatmap we can observe that

- ❖ is_smoking and cigsPerDay are 76% correlated with each other
- ❖ prevalent_hyp and sysBP are 70% correlated with each other
- ❖ sys_BP and diaBP are 78% correlated with each other
- ❖ prevalent_hyp and diaBP are 61% correlated with each other
- ❖ glucose and diabetes are 61% correlated with each other



Feature Engineering :

Converted the categorical features to Numerical features

- For ‘sex’ column, (Male = 0, Female = 1)
- For ‘is_smoking’ column (Yes = 1, No = 0)

For removing multicollinearity,

- sysBP and diaBP are 71% correlated with each other.
- sysBP means Systolic blood pressure which measures the pressure in your arteries when your heart beats.
- diaBP means Diastolic blood pressure which measures the pressure in your arteries when your heart rests between beats.
- The top number (systolic) minus the bottom number (diastolic) is the pulse pressure.
- We can create a new column ‘pulse_pressure’ by subtracting diaBP from sysBP.
- We can also drop ‘sysBP’, ‘diaBP’, ‘prevalentHyp’, ‘is_smoking’ features.

Removing Outliers :

- At first, created a ‘for’ loop for all the numerical features which appends the outlier index value into a variable.
- After storing the list of outlier indexes, simply dropped them from the data.

```
# Removing Outliers
# Creating a for loop for storing the indeces of outliers
for i in cont_feat:
  indeces = []
  x = data[i]
  mean = data[i].mean()
  std = data[i].std()
  index = data[(np.abs(x)) - (mean) >= (3 * std)].index
  indeces.append(index)
```

```
# Displaying the list of indeces of outliers
outliers_index = list(indeces)[0]
outliers_index
```

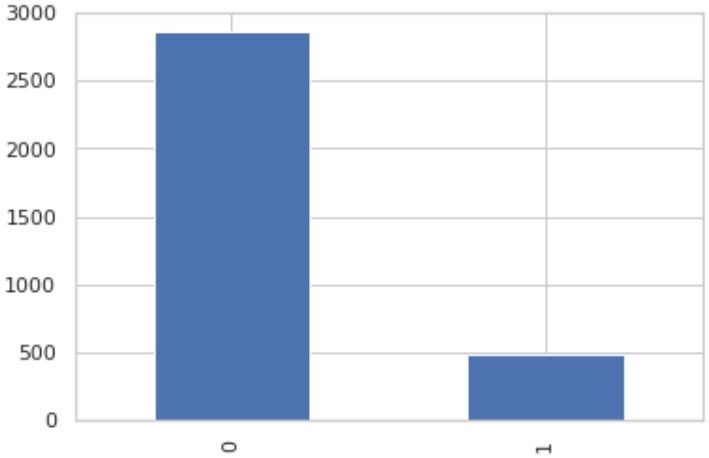
```
Int64Index([  77,  107,  134,  151,  173,  230,  312,  400,  481,  534,  594,
             786, 1032, 1150, 1156, 1281, 1559, 1587, 1610, 1819, 1947, 1977,
            1993, 2164, 2187, 2188, 2262, 2566, 2672, 2703, 2755, 2785, 3042,
            3045, 3063, 3069, 3092, 3117, 3164, 3232, 3260, 3373],
           dtype='int64')
```

```
# Removing outliers
df.drop(outliers_index, inplace = True)
```

Pre Processing:

- After converting the categorical features into numerical features, the data has only numerical values.
- Feature scaling is a important preprocessing step. So we can use StandardScaler for feature scaling.
- Features that are measured at different scales do not contribute equally to the model fitting & model learned function and might end up creating a bias.
- Thus, to deal with this potential problem feature-wise normalization such as StandardScaler Scaling is usually used prior to model fitting.
- After scaling the data, the data is ready to fit into model.
- We can split the data into Training dataset and Test dataset with test size of 20%.

Handling Class Imbalance of target feature :




```
# Lets observe the classes of target feature
df_scaled['TenYearCHD'].value_counts()
```

```
0    2859
1     489
Name: TenYearCHD, dtype: int64
```

- Here, we can see the count of '0' class is high which indicates the class imbalance.
- For handling class imbalance, we can use SMOTETEK which will resample the training dataset into equal classes.

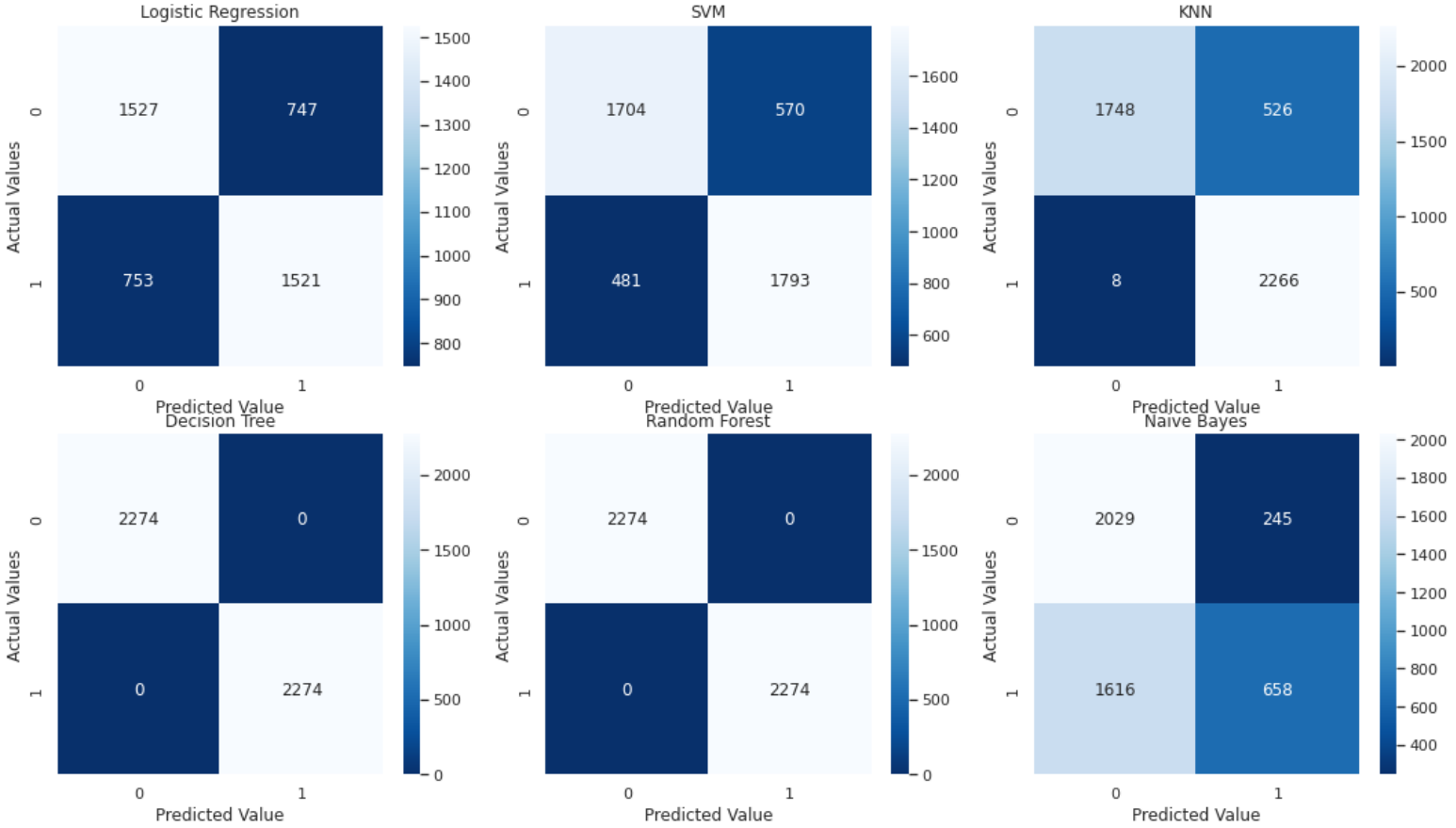
Model Implementation :

In order to create a model pipeline, I append the default state of all classification algorithms into the model list and then iterate through them to train, test, predict and evaluate. Those classification algorithms are

1. Logistic Regression
2. Decision tree
3. Random Forest
4. Support Vector Machine
5. K-Nearest Neighbour (KNN)
6. Naïve Bayes

- For classification task, there are various metrics used for evaluation.
- For our problem statement, we have two cases
- Case 1 : if a person has a disease but the model shows the person has no risk. For this case recall is the best evaluation metric
- Case 2 : if a person has no risk of disease but the models predicts the person has a risk. For this case, precision is the best evaluation metric
- When both of the above cases are important, we can use F1_score evaluation metric

Confusion Matrix of models for Train Data :

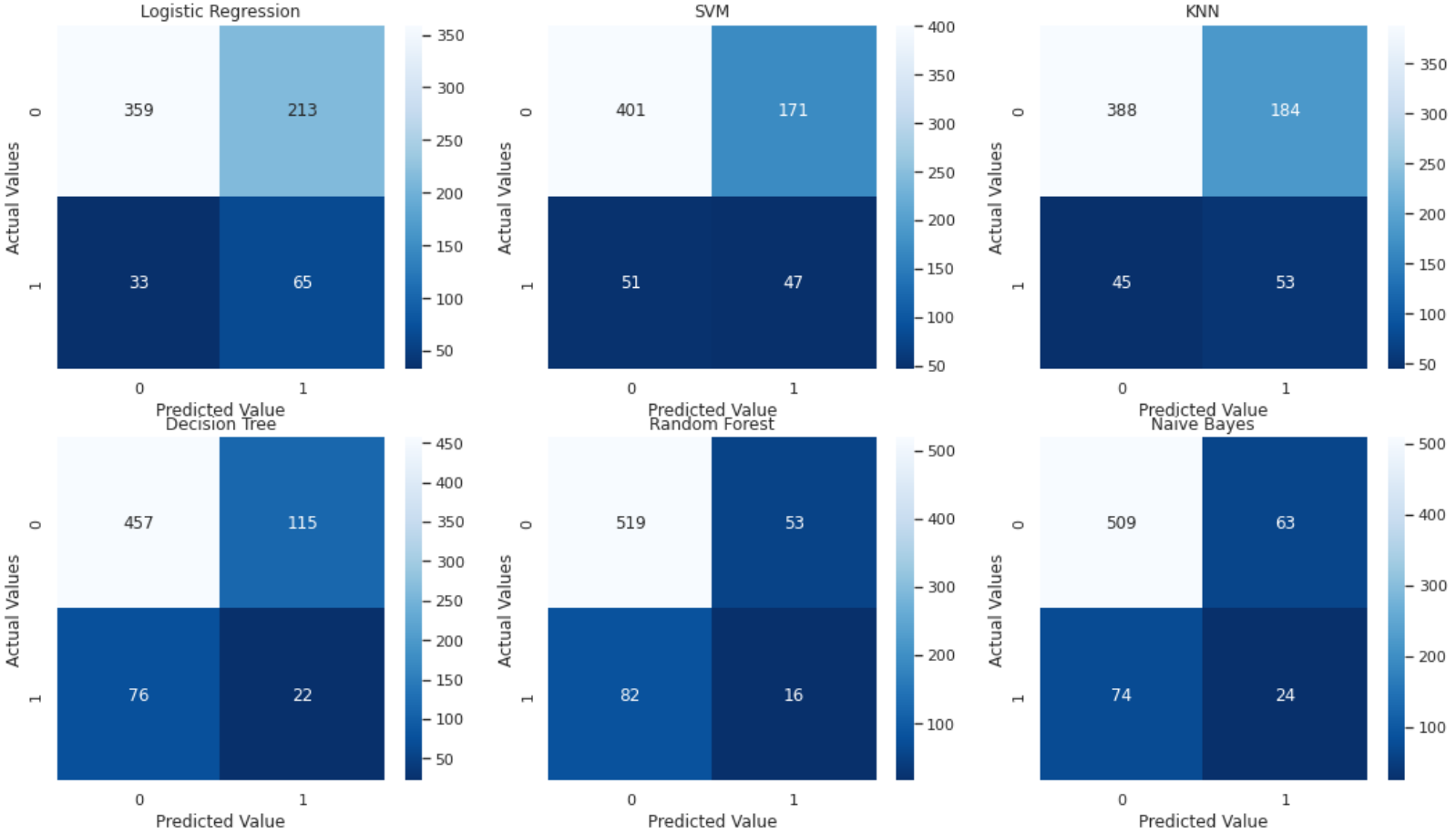


Evaluation Metrics for Train Data :

	Model	Accuracy	AUC	Precision	Recall	F1 Score
0	Logistic Regression	0.67018	0.67000	0.67063	0.66887	0.66975
1	SVM	0.76891	0.77000	0.75878	0.78848	0.77334
2	KNN	0.88259	0.88000	0.81160	0.99648	0.89459
3	Decision Tree	1.00000	1.00000	1.00000	1.00000	1.00000
4	Random Forest	1.00000	1.00000	1.00000	1.00000	1.00000
5	Naive Bayes	0.59081	0.59000	0.72868	0.28936	0.41423

From the above visualizations and dataframe, we can say that Decision Tree and Random Forest model are overfitted.

Confusion Matrix of models for Test Data :



Evaluation Metrics for Test Data :

	Model	Accuracy	AUC	Precision	Recall	F1 Score
0	Logistic Regression	0.63284	0.65000	0.23381	0.66327	0.34574
1	SVM	0.66866	0.59000	0.21560	0.47959	0.29747
2	KNN	0.65821	0.61000	0.22363	0.54082	0.31642
3	Decision Tree	0.71493	0.51000	0.16058	0.22449	0.18723
4	Random Forest	0.79851	0.54000	0.23188	0.16327	0.19162
5	Naive Bayes	0.79552	0.57000	0.27586	0.24490	0.25946

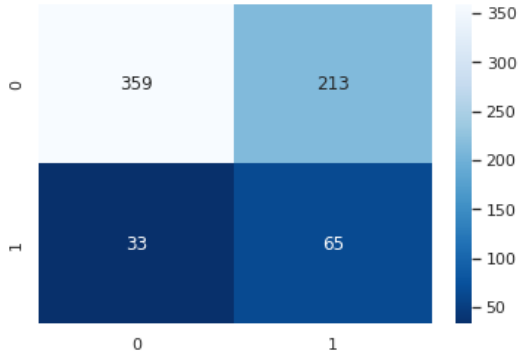
From the above metrics Data frame, we can say that

- **If Recall is important, i.e., if we want to focus on case 1, then Logistic Regression and KNN models are preferrable.**
- **If Precision is important i.e., if we want to focus on case 2, then Naive Bayes model is preferrable.**
- **If F_Score is important, i.e., if we want to focus on both case 1 and case 2, then Logistic Regression model is preferrable.**

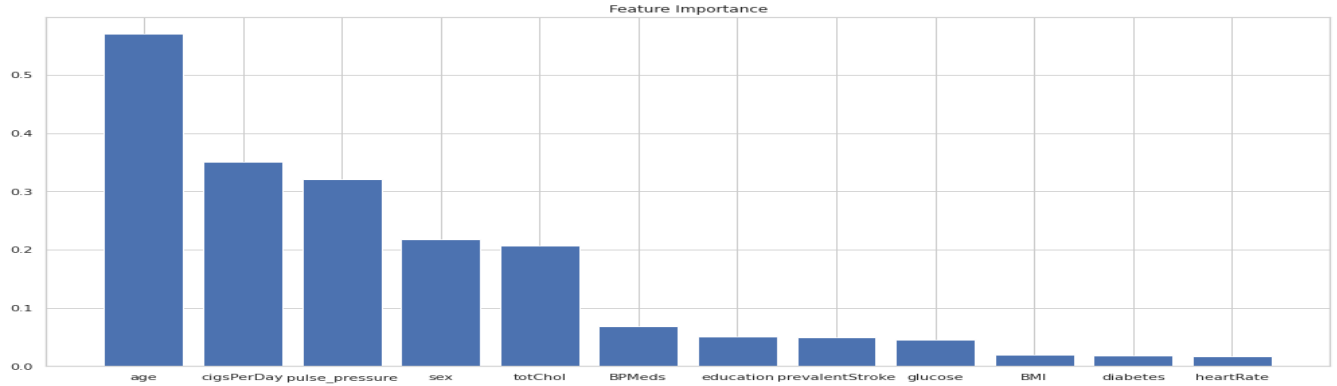
Logistic Regression with HyperParameter Tuning :

After Tuning the hyperparameters for logistic Regression Model, the best parameters are

{'C': 1, 'max_iter': 100, 'penalty': 'l2'}



	0	1	accuracy	macro avg	weighted avg
precision	0.91582	0.23381	0.63284	0.57481	0.81606
recall	0.62762	0.66327	0.63284	0.64544	0.63284
f1-score	0.74481	0.34574	0.63284	0.54528	0.68644
support	572.00000	98.00000	0.63284	670.00000	670.00000

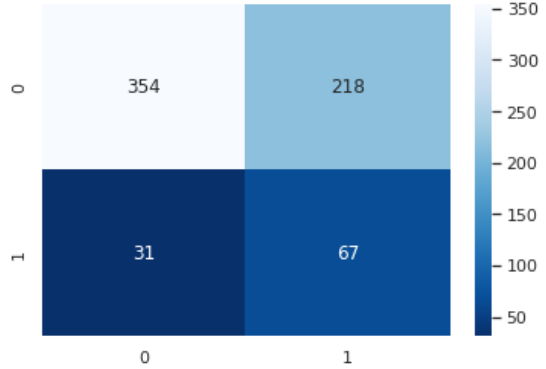


	Metric	score
0	Accuracy	0.63284
1	Precision	0.23381
2	Recall	0.66327
3	F1 Score	0.34574

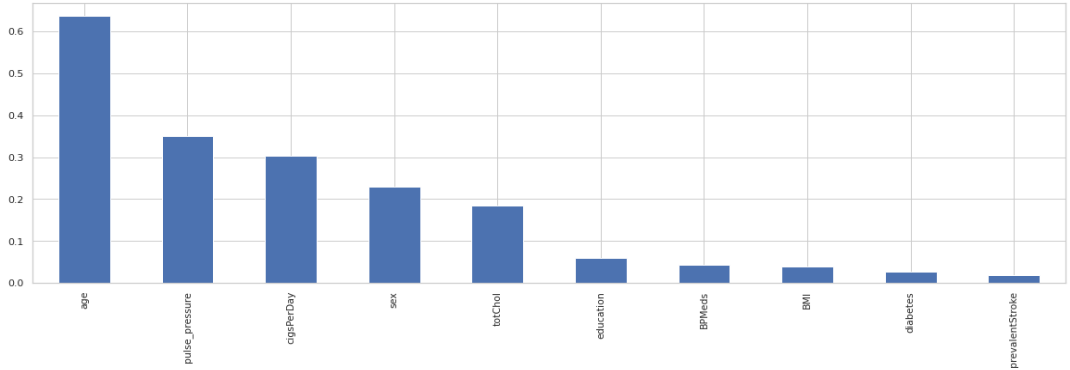
Support Vector Machine with HyperParameter Tuning :

After Tuning the hyperparameters for Support Vector Machine Model, the best parameters are

{ kernel = 'linear', C = 1 }



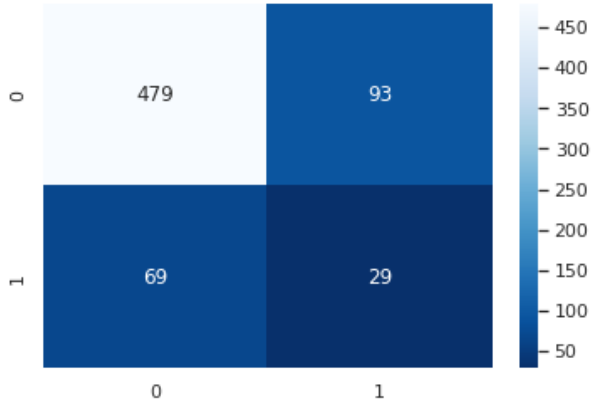
	0	1	accuracy	macro avg	weighted avg
precision	0.91948	0.23509	0.62836	0.57728	0.81938
recall	0.61888	0.68367	0.62836	0.65128	0.62836
f1-score	0.73981	0.34987	0.62836	0.54484	0.68278
support	572.00000	98.00000	0.62836	670.00000	670.00000



	Metric	score
0	Accuracy	0.62836
1	Precision	0.23509
2	Recall	0.68367
3	F1 Score	0.34987

KNN Classifier with HyperParameter Tuning :

After Tuning the hyperparameters for KNN Classifier Vector Model, the best parameters are {'n_neighbors': 2}



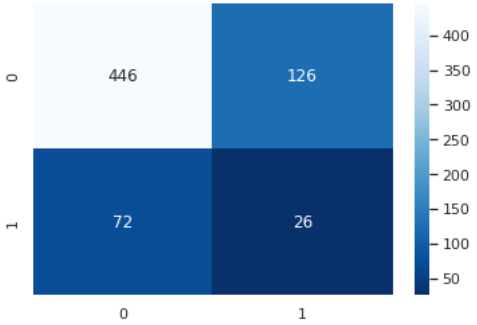
	0	1	accuracy	macro avg	weighted avg
precision	0.87409	0.23770	0.75821	0.55590	0.78100
recall	0.83741	0.29592	0.75821	0.56667	0.75821
f1-score	0.85536	0.26364	0.75821	0.55950	0.76881
support	572.00000	98.00000	0.75821	670.00000	670.00000

	Metric	score
0	Accuracy	0.75821
1	Precision	0.23770
2	Recall	0.29592
3	F1 Score	0.26364

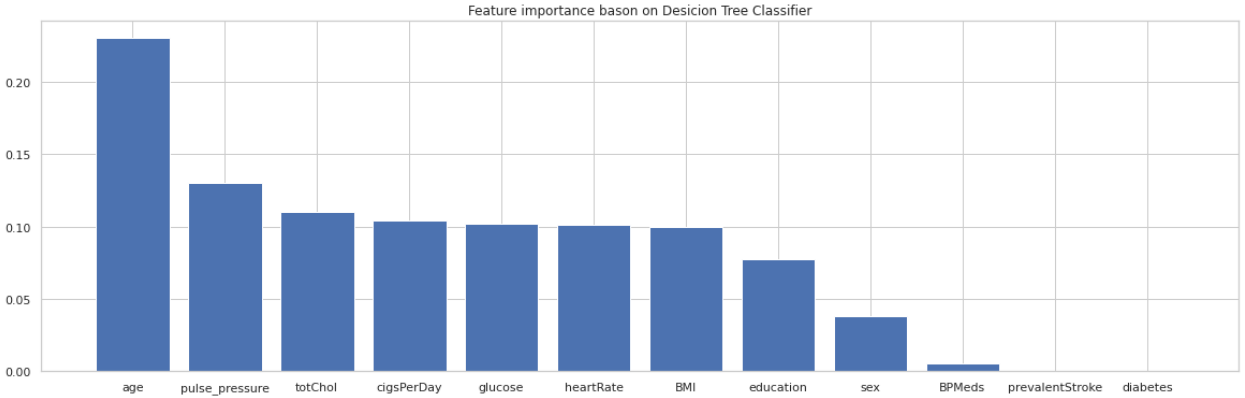
Decision Tree Classifier with HyperParameter Tuning :

After Tuning the hyperparameters for Decision Tree Classifier Model, the best parameters are

{'criterion': 'gini', 'max_depth': 9, 'min_samples_leaf': 1, 'min_samples_split': 6}



	0	1	accuracy	macro avg	weighted avg
precision	0.86100	0.17105	0.70448	0.51603	0.76009
recall	0.77972	0.26531	0.70448	0.52251	0.70448
f1-score	0.81835	0.20800	0.70448	0.51317	0.72907
support	572.00000	98.00000	0.70448	670.00000	670.00000

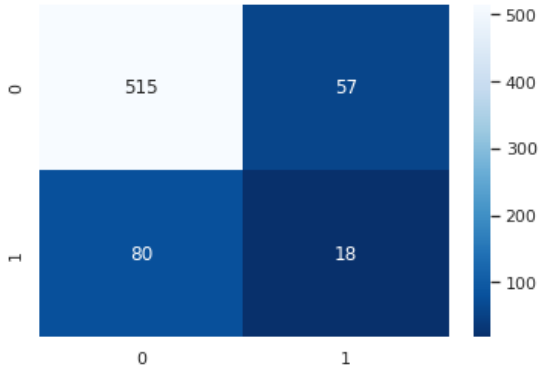


	Metric	score
0	Accuracy	0.70448
1	Precision	0.17105
2	Recall	0.26531
3	F1 Score	0.20800

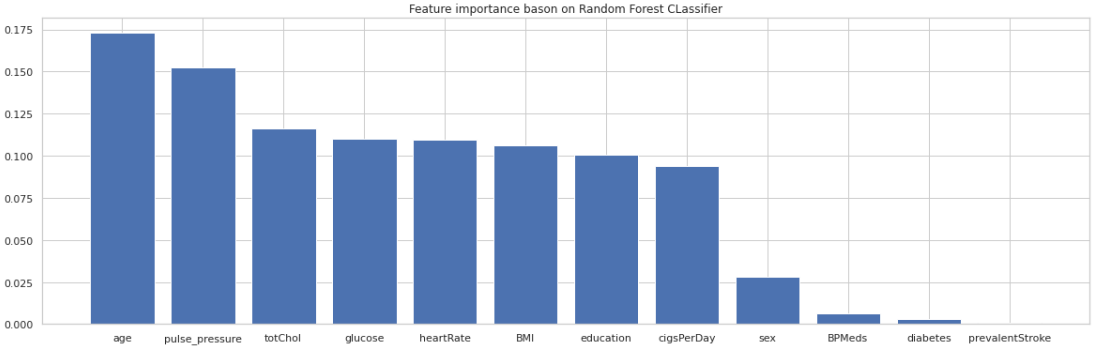
Random Forest Classifier with HyperParameter Tuning :

After Tuning the hyperparameters for Random Forest Classifier, the best parameters are

{'max_depth': 16, 'n_estimators': 256}



	0	1	accuracy	macro avg	weighted avg
precision	0.86555	0.24000	0.79552	0.55277	0.77405
recall	0.90035	0.18367	0.79552	0.54201	0.79552
f1-score	0.88260	0.20809	0.79552	0.54535	0.78394
support	572.00000	98.00000	0.79552	670.00000	670.00000



	Metric	score
0	Accuracy	0.79552
1	Precision	0.24000
2	Recall	0.18367
3	F1 Score	0.20809

Conclusion :

From Exploratory data Analysis, we can conclude that

- Almost half of the patients have a habit of smoking
- Outliers present in continuous features except age column.
- The patients who have more age, totChol, sysBP, diaBP, BMI, heartRate, glucose are tending towards risk of coronary heart disease
- 58.4 % of females and 41.5 % of males have no risk of CHD
- 46.7 % of females 53.22 % of males have risk of CHD
- 50.9 % of people who doesn't smoke have no risk of CHD
- 49 % of people who smoke have no risk of CHD
- 46.1 % of people who doesn't smoke have risk of CHD
- 53.8 % of people who smoke have risk of CHD

By fitting the data into various classification models and evaluating with test data, we can conclude that,

For our problem statement, we have two cases

- Case 1 : if a person has a disease but the model shows the person has no risk. For this case recall is the best evaluation metric

- Case 2 : if a person has no risk of disease but the models predicts the person has a risk. For this case, precision is the best evaluation metric
- When both of the above cases are important, we can use F1_score evaluation metric
- If Recall is important, i.e., if we want to focus on case 1, then Logistic Regression and KNN models are preferrable
- If Precision is important i.e., if we want to focus on case 2, then Naive Bayes model is preferrable
- If F_Score is important, i.e., if we want to focus on both case 1 and case 2, then Logistic Regression model is preferrable
- For Random Forest Classifier, 'age' feature has more importance and least is 'PrevalentStroke'.
- For SVM Classifier, 'age' feature has more importance and least is 'prevalentStroke'.
- For Decision Tree Classifier , 'age' feature has more importance and least is 'prevalentStroke', 'diabetes'.
- For logistic regression, 'age' feature has more importance and least is 'BMI', 'diabetes', 'heartRate'.

THANK YOU